MS & PC DOS
内存驻留、中断和磁盘管理

中国科学院希望高级电脑技术公司

MS & PC DOS

内存驻留、中断和磁盘管理

亦 欧 五 月 译
石 放 泰迪·道尔 校

中国科学院希望高级电脑技术公司

一九九0年八月

■ 北京市新闻出版局
准印证号： 3185—90185
■ 订购单位：北京 8721信箱资料部
■ 电　　话： 2562329
■ 电　　传： 01—2561057
■ 电　　挂： 0755
■ 地　　址：海淀影剧院北侧
■ 乘　　车： 320、332、302路至海淀黄庄下车
■ 办公地点：希望公司大楼 101 房间

目　录

第一部分　开发磁盘功能

在这个部分里，将对磁盘进行检查，将要研究导引记录、FAT、目录、子目录、文件以及分区。你将会了解它们的功能，如何使用它们以及不同类型磁盘的区别。我们将设计一个工具来探索和修改磁盘，并试验一些磁盘方面的技巧。最后一章以编写一个修改DOS命令名的程序结束。

了解磁盘是如何工作的能在多个方面对你有所帮助。假使用户在碰到了磁盘方面的问题，如偶然删掉了一个文件、重新格式化了硬盘、或者破坏了目录，就可以知道如何把数据找回来。下面的知识还能帮助用户开发或使用涉及磁盘的应用程序，如数据库或保密系统。

第一章　关于磁盘

每当使用计算机时，都要用到磁盘。从磁盘进行启动，从磁盘装入程序，结果也要保存到磁盘上。下面几章中，将要介绍信息在磁盘上是如何存贮和组织的，这有助于编写有效的工具来增强使用磁盘的效率。

通常，可以将磁盘看作是一个文件集合。用户可运行程序文件，编辑数据文件，或者对它们进行拷贝和删除，但这并不是全部。

实际上，磁盘上的信息非常之多，它需要知道这是什么类型的磁盘，其中包含了哪些文件，位于何处，哪里还有空间以容纳更多的文件。这些信息通常隐藏在用户看不到的四个基本结构中：分区表、导引记录、文件分配表以及目录。磁盘还有面、磁道和扇区等物理结构。首先让我们看一看磁盘的物理结构。

磁盘（软盘和硬盘）是由磁物质组成的圆盘片。硬盘是由一组盘片组成的。每张盘片有一个顶面和一个底面，磁盘的一面被称为面或头。头指的是实际进行读动作的电磁设备。磁盘上或多或少都有一些电磁体。软磁盘有两面，通常硬盘的每张盘片上有两面。

每一面又进一步分成许多称为磁道或磁道柱面的同心圆。磁头可移到磁盘的任一磁道上。当磁盘在磁头下旋转时，磁头就能够读取存贮在磁盘上的各种信息。

每个磁道又被分为扇区。扇区是对磁盘进行读写信息的最小单位，通常每个扇区为512字节长。

磁盘上要被管理的扇区太多，因此，扇区被组合成簇。簇只是几个连续扇区的集合。每簇可有二至八个扇区，这取决于磁盘的类型。图1-1给出了一张磁盘的示图。

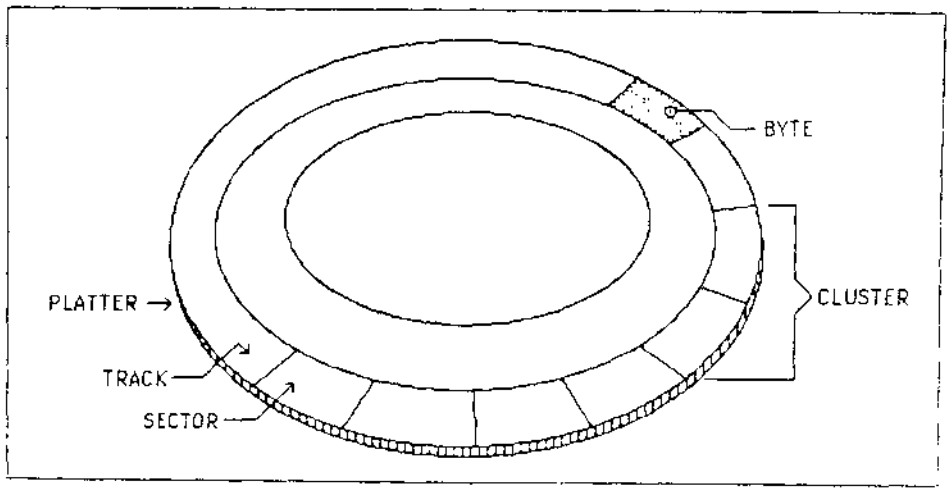

图1.1　磁盘

这听起来也许很复杂，但幸运的是DOS为用户做了大部分的工作。通常用户不必考虑簇和扇区，以及磁道和面。

让我们来看看说明磁盘是如何被组织的四个基本信息部分。先从分区表开始。多数情况下我们只使用一个操作系统（DOS），但有些用户要用到 CP／M 或 XENIX。如果这些用户有硬盘，他们当然希望能用它来存贮这些操作系统的文件，而它们使用磁盘的方法与 DOS 是不兼容的。为了将硬盘用于多个操作系统，可将硬盘分为至多四个分区，每一分区可以使用一种操作系统。

分区表用来记录这四个部分的每一部分的开始位置和长度，如果计算机从硬盘启动使用哪一部分（该部分称为活动分区），哪一部分就使用 DOS 作为操作系统，这一点是很重要的。在下面几章你将会了解到 DOS（以及实用程序）要在磁盘上的某个地方寻找某种类型的信息。在硬盘上，这些预先定义的位置可在任何部位，这取决于分区是如何设置的。DOS 使用分区表来了解使用的分区从何处开始并以此作为对磁盘访问的偏移量。这样，总能找到特定的信息，而且分区设置对 DOS 和用户来说都是透明的。

当打开计算机时，ROM 中的一个程序首先试图从 A 驱动器中的磁盘进行导引。如果 A 驱动器中没有磁盘，它就将控制交给分区表前的一个程序（如果系统没有硬盘，就启动 ROM BASIC），该程序读取分区表。如果其中的信息是有效的，它就找出活动分区的导引记录的位置并将控制传到那里。

记住，只有硬盘才有分区表。

导引记录是磁盘上第二个重要的部分。它包含了调入并启动操作系统的程序，还包含了一张描述磁盘的信息表：大小、类型、操作系统、格式、以及关于文件分配表和目录的重要信息。DOS 就用这些信息来确定如何对磁盘进行读写。

数据以簇为单位存贮在文件中。通常，这些簇不必一个紧接一个；文件的第一部分可以在磁盘开始处附近的某个部位，而文件的其余部分都在磁盘的中间或尾部。文件分配表（FAT）说明了每个文件使用哪些簇。它还说明了磁盘的哪些部分是空闲可用的，哪些部分因物理损伤而不可用。文件分配表很重要，故在磁盘上为它保存了两个拷贝。如果没有它，磁盘上的所有数据都只是没有意义的随机组织的字节片段而已。

目录是最后一个组织结构，它说明文件的名字、长度、建立时间，以及 DOS 要查找的信息。

§ 1.1 程序设计要点

- 磁盘由面、磁道、扇区组成。
- 扇区被组合为称为簇的组织单元。
- 磁盘的五个基本部分是分区表、导引记录、文件分配表、目录、数据区。

第二章　EXPLORER 概貌

让我们来详细了解磁盘是如何工作的。不仅要了解磁盘是如何组织和使用的，而且还要进行实际检验。为此要编写一个 Pascal 磁盘开发工具。在讨论了磁盘的各种特征后，要将相应的模块加到程序中以检验它们。图 2.1 给出了 EXPLORER 的主菜单。

```
-----------------------MAIN MENU---------------
F1:    Dump and Modify Sectors
F2:    Examine Boot Record
F3:    Cluster by Cluster FAT Dump
F4:    Examine Root Directory
F5:    Examine File
F6:    Get Back Erased File
F7:    Examine Partition

F10:   Exit
```

图 2.1　EXPLORER 的主菜单

本书有这个程序的所有代码，它是用 Turbo Pascal 3.0 编写的，如果你没有这个版本以上的 Pascal 版本，就需要对它作一些修改。也可以使用《高级 DOS 磁盘指南》，它包含了所有程序的源代码和可执行程序，故不必再敲入这些代码，同时也不必担心会有错误；可直接使用这些已经编译好的程序。如果你没有 Pascal 编译器，就必须要有这张磁盘。EXPLORER.COM 程序是极其有用的。在本书的各处都要用到它。注意保护好这片磁盘。

可以将 EXPLORER 做得更灵活，它是菜单驱动的全屏输入例程。主例程读取命令并将控制传给更具体的过程，后者又分别保存自己命令的子菜单。可使用正文窗口以保持屏幕整洁。

由于 EXPLORER 的代码很长，可以每次只写一部分，并用包括命令（$I）将该部分和主模块一起编译。主模块命名为 explorer.pas。

本章里要经常运行 EXPLORER，但在运行前可能要离开 Turbo Pascal 集成环境。因为 Turbo Pascal 不时要存贮磁盘信息，这就使 EXPLORER 的操作不正确了。可以将 EXPLORER 编译为.COM 文件，离开 Turbo Pascal 环境，然后运行它，这样它就能正确工作了。

本书中要用到二进制、十进制、以及十六进制。二进制数写为 100111b，十进制记为 14，而十六进制数形如$16 或 16h。

§2.1 程序设计要点

- 准备建立磁盘开发工具 EXPLORER。
- 把 EXPLORER 代码编写为一系列模块，用包括命令（$I）将每个模块加到 explorer.pas 中，然后再编译 explorer.pas。

第三章　命令获取和磁盘 I/O

我们将以 explorer.pas 为基础编写 EXPLORER。在 explorer.pas 中，将为整个程序设置常量和类型定义，并且处理主菜单命令。在 explorer.pas 中，还包括对磁盘读写的基本例程。

为了进行灵活的控制，需要编写自己的例程来从键盘读取命令和完成磁盘 I/O。磁盘 I/O 例程比下面四章中的任何一个例程都复杂，因为它们要用到汇编语言。

如果你不熟悉汇编语言程序设计，就会发现这些例程相当难懂。为了提高这些例程的效率，需要用 BIOS 和 DOS 中断。如果你不懂这些内容，就略过它们而直接使用本章后部的代码。你不必完全理解它，但至少必须知道它是干什么的。

§3.1　读取命令

为使 EXPLORER 易于使用，并且看起来专业化，就必须能处理功能键和光标键。不幸的是，readln 不能处理这些键，因此要使用 BIOS 中断（第二部分再深入讨论 BIOS 和 DOS 中断）。

BIOS 中断是存在 ROM 中的子程序，它执行计算机的基本硬件动作。例如，有的 BIOS 中断在屏幕上打印字符，有的检查光笔的位置，有的向打印机发送信息，还有的 BIOS 中断从键盘读取击下的键。每个中断都有一个中断号和一组使用的参数。

为了从 Pascal 调用中断，可用 intr 命令来传递中断号和参数。参数使用下列数据结构的集合：

```
result = record
  ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
end;
```

其中每个整型变量表示一个 8086 寄存器。有时，可以用 ah，al，bh，bl，ch，cl，dh，dl 来代替 ax，bx，cx，dx。这只是相应寄存器的高位或低位命令字节而已。例如 ah = ax div 256; al = ax mod 256。

用中断 16h 可读取键盘。以 AH=0 来调用该中断，该中断例程就等待击键，然后，在 AX 中返回击下的键码。如果 AL 非 0，该键就是一个普通的字母或数字键，并且 AL 包含了它的 ASCII 值。但如果 AL 为 0，AH 就包含了一个表示功能键或光标键的代码。

在第十五章中将详细讨论中断 16h 和它返回的代码。现在，先看一看使用中断 16h 的过程，以便了解它返回值的类型。例如，从主过程可看到，如果按下了 F10 键，中断就以 AX = 4400h 返回。

读取键盘的 Pascal 代码形式如下：

```
regs.ax := 0;                    {regs的类型为result。AX = 0表示读键}
intr($16,regs);                  {等待击键并返回键码}
if regs.ax and $ff=0 then...     {功能键或光标键}
```

可用下列语句来检查该值：

```
case (regs.ax) of...
```

explorer.pas 的代码列在本章末尾。可浏览一下其中读取键盘的例程。

§3.2 磁盘读写

对磁盘读写也要使用中断，但这回使用的是 DOS 中断。DOS 中断与 BIOS 中断很相似，只是 DOS 中断例程不是在 ROM 中，它是在导引计算机时装入的。但对我们的程序来说，这并没有什么区别。

可使用中断 25h 和 26h 来读写扇区。扇区通过逻辑扇区号引用。磁盘上的第一个扇区号数为 0，其次为 1，依此类推，直到没有剩余扇区为止。

有时，也可以通过物理扇区号来引用扇区，此时，需要指明面和磁道。每个磁道的扇区编号从 1 到磁道的最后一个扇区。DOS 中断使这项工作变得很容易，DOS 会考虑磁道和面的转换。只需给出一个扇区号，DOS 就帮你找到它（而 BIOS 磁盘中断使用的是面、磁道、扇区这一种引用方法）。

在对磁盘读写时几乎总是使用到逻辑扇区号。

中断 25h 以四个参数调用：

```
AL = 驱动器号（0 = A，1 = B，2 = C，等等）
CX = 要读的扇区号
DS:BX = 放置读入数据的位置    （DS 为缓冲区地址段；BX 为偏移量）
```

中断 26h 使用相同的参数，但 CX 是要写的扇区号，DS:BX 是要写出数据的位置。

如果仅需用 intr 命令来调用这些中断就好了，象读取键盘那样。不幸的是，这两个中断很难处理。和别的中断不一样的是，它们在回到调用程序之前并不恢复现场，它们在栈中留下了一个状态标志。如果使用 intr 命令，在进行读写的过程完成之前一切正常，但在试图返回调用它的过程时，它就会崩溃。

为解决这个问题，需要在 explorer.pas 中通过 inline 函数结合使用一些小的汇编语言程序。要对磁盘读写的部分详细检查。

建立一个名为 DiskRead 的函数来从磁盘读取数据。该函数的参数有驱动器号，开始扇区号，读取的扇区数，以及一个指向存贮信息的缓冲区的指针。如果不能从磁盘读，就返回一个错误代码。可使用 inline 代码来调用 DOS 中断，检查错误，清除并返回。

仔细注意 Turbo Pascal 和汇编语言的接口处理。它被不恰当地编入了 Turbo Pascal 手册（2.0 和 3.0 版）。

在调用一个函数时，它把其参数送入栈中。值参数作为数据传送；变量参数和数组作为指针传送。不同类型的数据以不同的格式传送。字节型（Byte）和整型（Integers）作为字（Words）传送；指针型（Pointers）以偏移量和段传送。其它类型的细节参见 Turbo Pascal 手册。

声明了函数后，函数结果所需的空间就被推入栈。对整型、布尔型、字节型和字符型来说，它是一个字。接着把函数声明中的参数从左到右推入栈。然后再推入三个字（这一点 Turbo Pascal 手册中没有记载），最后，调用 inline 代码。

为读取参数，首先将 bp 寄存器推入栈以保存其内容。然后，将 bp 设置为 sp 再加 8 以指向参数区（2 个是刚才推入的 bp，6 个是推入栈的三个字），增加 bp 就能得到下一个参数。记住，由于参数是由左到右推入栈的，在增加 bp 时就会以从右到左的顺序访问到它们。

对我们的函数来说，要传送一个字节型，两个整型值，以及一个指针，函数结果是整型值。因此，在将 bp 入栈，设其为 sp 值，再增加 8 之后将：

```
bp      ;将指向指针的偏移量
bp+2    ;将指向指针的段值
bp+4    ;将指向一个整数
bp+6    ;将指向一个整数
bp+8    ;将指向一个字节（作为字存贮）
bp+10   ;将指向函数结果
```

记住要考虑 inline 代码中推入栈的所有值。加 8 是由于推入了 bp，以及 Turbo Pascal 推入的三个字。

为返回函数结果，要在栈中对应返回值的位置放上希望返回的值。

在 inline 代码中可以修改 8086 的所有寄存器。在退出前，必须要恢复 bp，sp，ds，和 ss。

让我们来看一看从磁盘读取一个扇区的汇编语言程序。该程序读取参数，调用 DOS 磁盘读中断、检测错误、清栈，并返回错误代码（如果没有错误，则为 0）。注意：代码末尾不必使用 ret 指令，这由 Turbo Pascal 函数的 end 来完成。为在 Pascal 程序中使用这个汇编语言程序，可先汇编它，然后再从.LST 文件中读取汇编后的值。这在 explorer.pas 程序清单中已经完成了。

通过 DOS 中断 25h 从磁盘读入扇区。假设这一调用发生在函数

```
xxx(drive: byte; sector,no_sects: integer; var buffer:array)
    : integer
```

中。drive 为指示磁盘驱动器的数字，0 = A, 1 = B, ...。Sector 是起始扇区，no_sects 是要读入的扇区数，buffer 是数据用的磁盘缓冲区。

成功的返回值为零，否则返回一个数字指出错误号。

```
push    bp              ;保存bp寄存器
```

```
        mov     bp,sp           ;取栈指针
        add     bp,8            ;指针移向参数
        push    ds              ;保存ds寄存器
        mov     bx,[bp]         ;磁盘缓冲区偏移量
        mov     ax,[bp+2]       ;磁盘缓冲区段地址
        mov     ds,ax           ;指示中断用缓冲区
        mov     cx,[bp+4]       ;扇区数
        mov     dx,[bp+6]       ;起始扇区
        mov     al,[bp+8]       ;驱动器
        push    bp              ;保存bp以免被中断破坏
        int     25h             ;发出DOS磁盘读中断
        jnc     ok              ;无携带值说明操作成功
        popf                    ;失败，弹出磁盘标志
        pop     bp              ;恢复bp
        mov     [bp+10],ax      ;将错误代码返回
        jmp     cont
ok:     popf                    ;成功，弹出磁盘标志
        pop     bp              ;恢复bp
        xor     ax,ax           ;以0代替错误号作为函数
        mov     [bp+10],ax      ;值返回
cont:   pop     ds              ;恢复ds
        pop     bp              ;恢复bp并继续执行函数中的代码
```

§ 3.3 EXPLORER.PAS

既然已经能够读取键盘命令，并能对磁盘读写，现在就看看完整的 explorer.pas 代码。注意怎样用 inline 命令进入磁盘读写例程，还要注意命令读取行。

有许多常量与窗口位置有关，在以后的模块中要经常用到窗口。注意磁盘缓冲区被设置为足够大，它可以同时读入九个扇区，这样做的原因在第五章讨论。

同时还要注意用于指明被检测驱动器和扇区的全局变量，这样就能在模块中对某个扇区定位并在另一个模块中修改它。

```
{.....DISK EXPLORER.....
```

这是一个硬盘开发程序，含有分区、记录导引、FAT 和目录的解码(decoding)命令，数据扇区和目录的检查、修改命令，文件显示命令和恢复被删除文件命令。此处为全部代码的主体—即被编译的部分。由于代码很长，各段被分别存放在清单所例的磁盘文件中。

```
.....}
program DiskExplorer;
{
```